Amendments to the Claims:

The listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

Claim 1 (currently amended): A method for ~~loading~~ operating a software module [[into]] on a processor unit in a controller networked via a data bus in a ~~means of transport, where~~ vehicle, wherein i) the software module [[(7)]] is executable in a plurality of controllers ~~(1, 3, 5) and the controllers (1, 3, 5)~~ which interchange data via the data bus [[(8)]], ~~where the~~ ii) selection ~~regarding~~ of the controller ~~(1, 3, 5)~~ on which the software module [[(7)]] is ~~loaded~~ operated is made based on the ~~basis of the computation~~ available computational capacity of the controllers ~~(1, 3, 5)~~ which are currently in operation, and iii) each of the controllers can turn off the software module when a utilization level of its processor is high, and as soon as the software module has been turned off, the software module is to be started again on another controller; said method comprising: ~~where~~

~~a check is performed to determine whether and on which controller (1, 3, 5) the software module (7) is running, this check is performed in rotation, that is to say in particular time periods, and each of the controllers (1, 3, 5) can turn off the software module (7) when the processor utilization level is high,~~

~~where as soon as the software module (7) has been turned off the software module (7) is started again, and where the check to determine whether and on which controller (1, 3, 5) the software module (7) is running is performed by virtue of the software module (7) sending an appropriate identifier containing these data to the data bus (8) in rotation or upon request, and~~

~~where it is established which of the controllers (1, 3, 5) involved in the data bus (8) has the maximum free computation capacity, that is to say the lowest processor utilization level in relation to the processor clock frequency, and where this information is obtained by virtue of the controllers (1, 3, 5) involved sending in rotation or by means of a request~~

a software module in a controller on which said software module is running sending to the data bus, either cyclically or upon request, an appropriate identifier containing information indicating its operating status and the identity of the controller on which it is running;

checking cyclically to determine whether and on which controller the software module is running, based on said identifier; and

determining which of the controllers has the greatest free computation capacity, taking into account its processor clock frequency;

wherein said determining step is made by virtue of the controllers involved sending in rotation or by means of a request, information that is indicative of their available computational capacity.

Claim 2 (currently amended): The method as claimed in claim 1, ~~characterized in that~~ wherein before the software module [[(7)]] is executed it is ascertained which of the ~~further~~ controllers ~~(1, 3, 5)~~ provides the maximum free computation capacity and the software module [[(7)]] is started on [[this]] the determined controller ~~(1, 3, 5)~~.

Claim 3 (currently amended): The method as claimed in ~~claims 1 and 2, characterized in that~~ claim 1, wherein the controller ~~(1; 3; 5)~~ on which the software module [[(7)]] is running compares its computation capacity with the computation capacity of the other controllers ~~(1; 3; 5)~~ and either continues to operate or terminates operation of the software module ~~(7) on the basis of~~ based on the comparison.

Claim 4 (currently amended): The method as claimed in ~~claims 1 to 4, characterized in that~~ claim 1, wherein the computation capacity of a controller ~~(1, 3, 5)~~ is ascertained from the processor utilization level and [[the]] processor type.

Claim 5 (currently amended): The method as claimed in ~~claims 1 to 5, characterized in that~~ claim 1, wherein the software module [[(7)]] is started on [[the]] a controller ~~(1, 3, 5)~~ having the maximum free computation capacity.

Claim 6 (currently amended): The method as claimed in ~~claims 1 to 6, characterized in that~~ claim 1, wherein the software module [[(7)]] is stored in [[the]] a memory [[means]] in the controllers ~~(1, 3, 5)~~.

Claim 7 (currently amended): The method as claimed in ~~claims 1 to 7, characterized in that~~ claim 1, wherein:

an identifier for the software module [[(7)]] is sent to the data bus ~~(8) in rotation~~ cyclically or upon request[[,]]; and

the identifier ~~containing~~ contains information about [[the]] an operating state and the operating controller ~~(1; 3; 5)~~ of the software module [[(7)]].

Claim 8 (currently amended): A networked controller having software modules ~~(2, 4, 6)~~ stored in ~~the microcontroller's~~ a controller's memory[[,]]; ~~where the~~ wherein:

the software modules ~~(2, 4, 6)~~ perform [[the]] primary control tasks; ~~of the relevant controller (1, 3, 5), where~~

a software module [[(7)]] with a subsidiary task can be additionally stored in a microcontroller's memory by the controllers; ~~(1, 3, 5), where~~

the controllers ~~(1, 3, 5)~~ have process cycles~~, where~~;

a process cycle is terminated after a particular time has elapsed, the data ascertained in the process are output onto the data bus [[(8)]], and the process cycle is started again;~~, where~~

the process cycle for the controllers ~~(1, 3, 5)~~ is determined by the software modules ~~(2, 4, 6)~~ for one of the primary task, ~~and/or~~ the operating system ~~and/or the~~ and a bus protocol[[,]]; and

[[where]] when a process cycle or a process cycle time has elapsed, data are sent to the data bus [[(8)]] which characterize their current processor utilization level and processor type used, with the controllers ~~(1, 3, 5)~~ using these data to ascertain the utilization level of the other controllers ~~(1, 3, 5)~~.

Claim 9 (new): A method of operating a network of controllers which are coupled via a data bus, each of which controllers has at least one processor, and has installed thereon the same software module which can be executed by the processor contained in any one of the controllers, each of said controllers being configured such that it can turn off the software module when a utilization of its processor is high, said method comprising:

each controller sending via the data bus, information regarding a current utilization level of its at least one processor;

whenever said software module is running in a particular one of said controllers, said software module in said particular controller sending via the data bus, an identifier indicating its operating state and identifying the particular controller;

checking said data bus to determine whether an identifier is present;

if no identifier is found in said checking step, determining which of the controllers has the greatest available computation capacity, based on its current utilization level as sent via the data bus;

said controller with said greatest available utilization level starting operation of said software module, and said software module sending to said data bus, an identifier indicating its operating status and the identity of the controller in which it is running;

if an identifier is present on the data bus in said checking step, the controller on which said software module is running ascertaining its own processor utilization level and comparing its computation capacity with the available computational capacity of other controllers coupled via the data bus;

if the utilization level of the controller on which the software module is greater than that of one of said other controllers, said controller on which said software module is running ceasing operation of said software module; and

said one of said other controllers starting operation of said software module, and said software module sending to said data bus an identifier indicating that it is running and identifying said one controller.